2. 4. Macros

The following macros are in macro.h

2.4.1. Constant Value

```
#define SOURCE              0
#define TEMPLATE            1

#define NO_ABS_POS          0
#define REF_ABS_POS         1
#define REAL_ABS_POS        2

#define FROM_NOWHERE                0
#define FROM_M_PFIRSTCHAIN          1
#define FROM_CARD                   2
#define FROM_M_PFIRSTDELCHAIN       3
```

2.4.2. INIT

2.4.2.1. INIT_STACK

```
#define INIT_STACK(stack)\
    stack.pFirstStatement = NULL;\
    stack.pLastStatement = NULL;
```

2.4.2.2. INIT_CHAIN

```
#define INIT_CHAIN(Chain)\
    Chain.bIsApplied = NULL;
```

2.4.3. FREE

```
#define FREE(pFirst, p)\
    while(pFirst){\
        p = pFirst->pNext;\
        free(pFirst);\
        pFirst = p;\
    }\
    p = pFirst;
```

2.4.3.1. FREE_STACK

```
#define FREE_STACK(stack)\
    FREE(stack.pFirstStatement, pStatement)
```

2.4.4. NEW

2.4.4.1. NEW_PAGE

```
#define NEW_PAGE(pPage, iChainBasePage)\
    pPage = new Page;\
    pPage->iPage = iChainBasePage;\
    pPage->pFirstCard = NULL;\
    pPage->pLastCard  = NULL;\
    pPage->pRootTmpVar= NULL;\
    pPage->pRootSrcVar= NULL;\
    pPage->pPrev      = NULL;\
    pPage->pNext      = NULL;
```

2.4.4.2. NEW_CARD

```
#define NEW_CARD(pCard, iChainBaseCard)\
    pCard = new Card;\
    pCard->iCard = iChainBaseCard;\
    pCard->iEntry = -1;\
    pCard->pFirstChain  = NULL;\
    pCard->pLastChain   = NULL;\
    pCard->pFirstUnit   = NULL;\
    pCard->pLastUnit    = NULL;\
    pCard->pPrev        = NULL;\
    pCard->pNext        = NULL;
```

2.4.4.3. NEW_ELEMENT

```
#define NEW_ELEMENT(pElement, Ele)\
pElement = new Element;
pElement->pPrev = NULL;
pElement->pNext = NULL;
pElement->pFirstAttr = NULL;
pElement->pLastAttr = NULL;
pElement->pChildChain = NULL;
pElement->bIsChainBase = false;
pElement->Ele = Ele;
pElement->Ele.bIsAbsPosOrg = Ele.cAbsPos;
pElement->Ele.bIsChanged = FASLE;
```

2.4.4.4. NEW_CHAIN

```
#define NEW_CHAIN(pChain, targetEle)\
pChain = new Chain;\
NEW_ELEMENT(pChain->pChainBase, targetEle);\
pChain->pFirstElement = NULL;\
pChain->pLastElement = NULL;\
pChain->pPrev = NULL;\
pChain->pNext = NULL;
```

Macro NEW_CHAIN is used in private method NewChain, NewChildChain, NewDivChildChain.

2.4.4.5. NEW_CHILD_CHAIN

```
#define NEW_CHILD_CHAIN(pChain)\
pChain = new Chain;\
pChain->bIsApplied = false;\
pChain->pChainBase = NULL;\
pChain->pFirstElement = NULL;\
pChain->pLastElement = NULL;
pChain->pPrev = NULL;\
pChain->pNext = NULL;
```

2.4.4.6. NEW_UNIT

```
#define NEW_UNIT(pUnit, pElement)\
pUnit = new Unit;\
pUnit->pElement = pElement;\
pUnit->pPrev = NULL;\
pUnit->pNext = NULL;\
```

2.4.4.7. NEW_FAMILY

```
#define NEW_FAMILY(pFamily, pUnit)\
pFamily = new Family;\
pFamily->iFamilyId = pUnit->pElement->Ele.iFamilyId;\
pFamily->pFirstUnit = pFamily->pLastUnit = pUnit;\
pFamily->pPrev = pFamily->pNext = NULL;
```

2.4.4.8. NEW_VAR

```
#define NEW_VAR(pVar, sCurrFrame)\
    pVar                = new Var;\
    pVar->iMaxFrame     = 0;\
    pVar->iMaxIFrame    = 0;\
    pVar->sFrame        = sCurrFrame;\
    pVar->bToOutput     = false;\
    pVar->pPrev      = NULL;\
    pVar->pNext      = NULL;\
    pVar->pFirstFrame   = NULL;\
    pVar->pLastFrame    = NULL;\
    pVar->pFirstIFrame= NULL;\
    pVar->pLastIFrame   = NULL;
```

2.4.5. APPEND

```
#define APPEND(pList, member, pItem, ItemName)
if(pList->member##pLast##ItemName)
    pList->member##pLast##ItemName->pNext = pItem;
else
    pList-> member##pFirst##ItemName=pItem;
pItem->pPrev = pList-> member##pLast##ItemName;
pItem->pNext = NULL;
pList->member##pLast##ItemName = pItem;

#define APPEND_M(pList, member, pItem, ItemName)\
if(pList->member##pLast##ItemName)\
    pList->member##pLast##ItemName->pNext = pItem;\
else\
    pList-> member##pFirst##ItemName=pItem;\
pItem->pPrev = pList-> member##pLast##ItemName;\
pItem->pNext = NULL;\
pList->member##pLast##ItemName = pItem;
```

2.4.5.1. APPEND_PAGE

```
#define APPEND_PAGE(this, pPage)\
APPEND_M(this, m_, pPage, Page)
```

2.4.5.2. APPEND_CARD

```
#define APPEND_CARD(pPage, pCard)\
APPEND(pPage, pCard, Card)
```

2.4.5.3. APPEND_ELEMENT

```
#define APPEND_ELEMENT(pChain, pElement)\
APPEND(pChain, pElement, Element)
```

2.4.5.4. APPEND_CHAIN

```
#define APPEND_CHAIN(pChain)\
APPEND_M(this, m_, pChain, Chain)
```

2.4.5.5. APPEND_CHAIN_TO_CARD

```
#define APPEND_CHAIN_TO_CARD(pCard, pChain)\
APPEND(pCard, pChain, Chain)
```

2.4.5.6. APPEND_DEL_CHAIN

```
#define APPEND_DEL_CHAIN(pChain)\
APPEND_M(this, m_, pChain, DelChain)
```

2.4.5.7. APPEND_ATTR

```
#define APPEND_ATTR(pElement, pAttr)\
APPEND(pElement, pAttr, Attr)
```

2.4.5.8. APPEND_UNIT

```
#define APPEND_UNIT(pCard, pUnit)
APPEND(pCard, pUnit, Unit);
```

2.4.5.9. APPEND_FAMILY

```
#define APPEND_FAMILY(pCard, pFamily)\
APPEND(pCard, pFamily, Family)
```

2.4.5.10. APPEND_UNIT_TO_FAMILY

```
#define APPEND_UNIT_TO_FAMILY(pCard, pUnit)\
bFamilyIsFound = false;\
for(pFamily = pCard->pFirstFamily; pFamily; pFamily = pFamily->pNext){\
    if(pFamily->iFamilyId == pUnit->pElement->Ele.iFamilyId){\
        APPEND(pFamily, pUnit, Unit);\
        bFamilyIsFound = true;\
        break;\
    }\
}\
if(bFamilyIsFound == false){\
    NEW_FAMILY(pFamily, pUnit);\
    APPEND_FAMILY(pCard, pFamily);\
    APPEND(pFamily, pUnit, Unit);\
}
```

2.4.6. STATEMENT

2.4.6.1. PUSH_STATEMENT

```
#define PUSH_STATEMENT(Stack, pStatement)\
if(Stack.pLastStatement)\
    Stack.pLastStatement->pNext = pStatement;\
else\
    Stack.pFirstStatement=pStatement;\
pStatement->pPrev = Stack.pLastStatement;\
pStatement->pNext = NULL;\
while(pStatement->pNext != NULL)\
    pStatement=pStatement->pNext;\
Stack.pLastStatement = pStatement;
```

2.4.6.2. POP_STATEMENT

```
#define POP_STATEMENT(Stack, pStatement)\
pStatement = Stack.pLastStatement;\
while(pStatement->bNewAction == false)\
    pStatement = pStatement->pPrev;\
Stack.pLastStatement = Stack.pLastStatement->pPrev;\
if(Stack.pLastStatement)\
    Stack.pLastStatement->pNext = NULL;\
else\
    Stack.pFirstStatement = NULL;
```

2.4.7. IS

2.4.7.1. IS_DESCENDANT

Decide whether Element Ele_2 is the descendant of Element Ele_1.

```
#define IS_DESCENDANT(Ele_1, Ele_2)\
    (Ele_1.iFamilyId == Ele_2.iFamilyId &&\
        Ele_1.sPath.substring(Ele_2.sPath)
```

2.4.7.2. IS_EQUAL

Decide whether Element Ele_2 equals Element Ele_1.

```
#define IS_EQUAL(Ele_1, Ele_2)\
    (Ele_1.iFamilyId == Ele_2.iFamilyId &&\
    Ele_1.sPath.equals(Ele_2.sPath))
```

2.4.8. INSERT

p1 and p2 are two pointers.

2.4.8.1. INSERT_BEFORE

Insert the structure instance pointed by p1 before the one pointed by p2.

```
#define INSERT_BEFORE(pFirst, p1, p2)
if(p2 == pFirst)\
    pFirst = p1;\
else\
    p2->pPrev->pNext= p1;\
\
p1->pPrev = p2->pPrev;\
p1->pNext=p2;\
p2->pPrev=p1;
```

2.4.8.2. INSERT_AFTER

Insert structure instance pointed by p2 after the one pointed by p1

```
#define INSERT_AFTER(pLast, p1, p2)
if(p1 == pLast)\
    pLast = p2;\
else\
    p1->pNext->pPrev= p2;\
p2->pNext = p1->pNext;\
p2->pPrev=p1;\
p1->pNext=p2;\
```

2.4.9. CUT

```
#define CUT(pFirst, pLast, pCurrent)\
if(pCurrent == pFirst){\
    pFirst = pCurrent->pNext;\
}\
else{\
    if(pCurrent == pLast){\
        pLast = pCurrent->pPrev;\
        pLast->pNext = NULL;\
    }\
    else{\
        pCurrent->pPrev->pNext = pCurrent->pNext;\
        pCurrent->pNext->pPrev = pCurrent->pPrev;\
    }\
}\
pCurrent->pPrev = pCurrent->pNext = NULL;
```

2.4.9.1. CUT_ELEMENT

```
#define CUT_ELEMENT(pFirst, pLast, pCurrent)\
    CUT(pFirst, pLast, pCurrent)\
    pCurrent->bIsChainBase = false;
```

2.4.10. REPLACE

```
#define REPLACE(pFirst, pLast, pOld, pNew)\
if(pOld == pLast){\
    pLast = pNew;\
}\
if(pOld == pFirst){\
    pFirst = pNew;\
}\
if(pOld->pPrev)\
    pOld->pPrev->pNext = pNew;\
if(pOld->pNext)\
    pOld->pNext->pPrev = pNew;\
pNew->pNext = pOld->pNext;\
pNew->pPrev = pOld->pPrev;\
pOld->pPrev = pOld->pNext = NULL;
```